gLExec and MyProxy integration in the ATLAS/OSG PanDA Workload Management System

J. Caballero[1], J. Hover[1], M. Litmaath[2], T. Maeno[1], P. Nilsson[3], M. Potekhin[1], T. Wenaus[1], X. Zhao[1]

[1] Brookhaven National Laboratory, PO BOX 5000 Upton, NY 11973, USA
[2] CERN, CH-1211 Geneva 23, Switzerland
[3] University of Texas, 701 S. Nedderman Drive Arlington, TX 76019, USA

E-mail: jcaballero@bnl.gov

Abstract.

Worker nodes on the grid exhibit great diversity, making it difficult to offer uniform processing resources. A pilot job architecture, which probes the environment on the remote worker node before pulling down a payload job, can help. Pilot jobs become smart wrappers, preparing an appropriate environment for job execution and providing logging and monitoring capabilities. PanDA (Production and Distributed Analysis), an ATLAS and OSG workload management system, follows this design. However, in the simplest (and most efficient) pilot submission approach of identical pilots carrying the same identifying grid proxy, end-user accounting by the site can only be done with application-level information (PanDA maintains its own end-user accounting), and end-user jobs run with the identity and privileges of the proxy carried by the pilots, which may be seen as a security risk. To address these issues, we have enabled PanDA to use gLExec, a tool provided by EGEE which runs payload jobs under an end-user's identity. End-user proxies are pre-staged in a credential caching service, MyProxy, and the information needed by the pilots to access them is stored in the PanDA DB. gLExec then extracts from the user's proxy the proper identity under which to run. We describe the deployment, installation, and configuration of gLExec, and how PanDA components have been augmented to use it. We describe how difficulties were overcome, and how security risks have been mitigated. Results are presented from OSG and EGEE Grid environments performing ATLAS analysis using PanDA and gLExec.

1. Introduction

The LHC (Large Hadron Collider)[1] is the next-generation particle accelerator/collider located at CERN (European Organization for Nuclear Research). It has been constructed by an international consortium in a 17-mile circumference tunnel beneath Switzerland and France. Four experiments have built massive detectors at four locations on the accelerator ring to record the showers of sub-atomic particles that result from collisions. ATLAS (A Toroidal LHC ApparatuS)[2] is one of these experiments.

The ATLAS detector, like the other LHC detectors, produces raw data in such quantities that storage and processing on site at CERN is impractical. Consequently, ATLAS has adopted a multi-tiered data storage and analysis model based on distributed, Grid computing[3]. Raw data is divided up and transferred via the Internet to several Tier-1 sites around the world, where it is placed on disk and stored permanently to tape. There, and at Tier-2 (and Tier-3)

sites it undergoes initial and subsequent rounds of processing and analysis on large computing clusters (typically rack-mounted commodity Linux systems).

All these computing sites are affiliated with Grids. In Europe, both the EGEE (Enabling Grids for E-sciencE)[4] and NDGF (Nordic DataGrid Facility)[5] projects provide the software and administrative infrastructure for making this distributed computing possible. In the U.S., OSG (Open Science Grid)[6] serves this purpose. Together, EGEE, NDGF and OSG have coordinated their activities in service of the LHC experiments' needs under the WLCG (Worldwide LHC Computing Grid) organization.

The ATLAS processing and analysis tasks are defined as distinct jobs, which are submitted to dozens of computing sites via their Grid interfaces. In this model, it is critical that jobs be dispatched to the location where the data on which it will operate already resides, since data transfer is very bandwidth-intensive while job submission is not.

To meet ATLAS requirements for a data-driven workload management system for production and distributed analysis processing capable of operating at LHC data processing scale the PanDA (Production and Distributed Analysis) system has been developed built upon the pilot-based framework, as described in [7]. It includes an important subsystem (the pilot scheduler) that manages the delivery of pilot jobs to worker nodes via a number of scheduling systems. Once launched on a worker node (WN), the pilot process contacts the job dispatcher and receives an available job matched to the site resources and characteristics, combined with brokerage policies.

PanDA was initially developed for US based ATLAS production and analysis, and assumed that role in late 2005. Since September 2006 PanDA has also been a principal component of the US Open Science Grid (OSG) program in just-in-time (pilot-based) workload management. In October 2007 PanDA was adopted by the ATLAS Collaboration as the sole system for distributed processing production across the Collaboration.

Despite the clear advantages of a pilot jobs architecture, making the working environment more homogeneous and isolating the job execution from the potential heterogeneities, there is an inherent security risk in the fact that the jobs inherit the identity of the user who originally submitted the pilot jobs to target sites. Accordingly, all end-user jobs will then possess the same identity and the same privileges granted by the proxy [8] carried by the pilots.

To address this security risk, the PanDA system has been given the capability to change the user identity of the payload job (initially run as pilot) such that it corresponds to that of the original job submitter. The change takes places based on the end-user credentials, previously stored on a caching service from where they are retrieved by the pilot process running on the WN. The mechanics of the identity switch are performed by gLExec. gLExec[9] is a super-user privileged executable with the capability of modifying the UID and GID to provide for a mapping between the grid user and the local Unix user accounts. This mapping is performed based on the results from gLite LCAS and LCMAPS[10] security components. The component that we use to perform the caching of the credentials is called MyProxy [11]. It is an open source application for managing grid proxy credentials. The identity change, as described above, is an optional feature of PanDA, which is employed only at sites which mandate its use. Such a condition is reflected in the site metadata recorded in the PanDA server's database.

2. Integration

2.1. Handling of the users proxies

The user proxies are handled in the following manner: in the distributed analysis scenario, when the user submits a job to the PanDA system, the client software (pathena)[12] will check if the user already has a proxy deposited on a dedicated MyProxy server, owned and managed by the ATLAS organization, and that this instance has a sufficiently long remaining period of validity (which is configurable). If that is not the case, the client will deposit a new user proxy onto the MyProxy server. The users credentials are never cached in PanDA.

When this new proxy is delegated to the MyProxy server, the identity of the entities authorized to retrieve it has to be specified. These identities are declared as a list of Distinguished Names (DN) corresponding to the pilot job submitters. In this way it is assured no other users than the authorized pilots will be allowed to retrieve the users' proxies from the server. This list of authorized retrievers is stored in the PanDA database, and read every time a new proxy is delegated. This DN list is expected to be superseded by just the VOMS pilot role when the MyProxy server supports this retrievers authentication and authorization mechanism.

In this scenario, the pilot job (and the pilot job only) has the credentials to extract the concrete user's proxy from the MyProxy server. This happens once the pilot connects to the PanDA server and is ready to execute the end user job. Finally, to be able to perform the UID and GID switch, the pilot identity must to be included in a super-user owned white list of users allowed to invoke gLExec.

Working under the assumption that we limit the lifetime of the user's proxy, the pilot job is responsible for its periodic renewal during the job's execution.

2.2. Security concerns and their mitigation

Possible security concerns related to misuse of a compromised pilot proxy are being addressed as follows. An instance of the proxy being stored on the MyProxy server can be assigned a unique key that will be required upon future retrieval. The key is stored on the PanDA server, and is delivered to the pilot when the latter obtains a new job from the server. In the interest of security, the key itself is a generated random string.

In addition, single (or few) use tokens will be used by the pilot job in order to get a payload job from the server. The pilot will have to present the token to the PanDA server in the job request process, and server has to match it to the value obtained from the database, where it was stored during the pilot submission process. After validating the pilot, the server immediately deletes the token (or decrements a maximum usage count).

The scheme described above helps to mitigate the few possible ways for an intruder to take possession of a user's proxy.

(i) To gain access to the PanDA database. We consider this level of security breach highly unlikely as it involves intrusion of a few loosely coupled components, each with its own set of credentials, and would require that all of these be simultaneously compromised. The access to the database is closely guarded, and in addition, as described above, there is a pre-requisite of having a valid pilot proxy in order to extract the user proxy.

(ii) To directly penetrate the Worker Node and impersonate pilot jobs. In case a Worker Node is compromised during a limited period (say a few hours or days), the attacker might try and imitate the pilot(s), requesting a series of jobs to obtain their associated user proxies, but the constraints on the token reuse ensure that for a given pilot any such abuse can only affect a small number of users. The attacker might kill the pilot itself, so that another job with another proxy (and possibly a token) gets started, but such an attack in itself is not specific to pilot-based workload management systems and thus is amenable to standard security measures. A similar scenario is one of abnormal job termination (such as when killed by the batch system or due to a power cut), whereby the proxy cached on the worker node is not immediately destroyed. Note, however, that to gain access to such a proxy one needs to possess the identity of the pilot job, which reduces the likelihood of such threats.

(iii) To gain direct access to the pilot job scheduler. This kind of threat is not unique in that there are already components in the grid infrastructure which, if compromised, could provide the intruder with credentials of multiple users. Such a risk is minimized by having a very small number of machines performing this task, together with establishing a strict control over access to these. Indeed, this is exactly the current practice.

3. Execution

To help define and secure the privileges and access rights of the pilot job, we are using the VOMS system [13]. The pilot job is started by a privileged user (who has the VOMS pilot role annotation), whose rights in this case are limited to being able to utilize gLExec. This is achieved by mapping the user (based on the aforementioned VOMS annotation contained in the proxy) to an account for which gLExec is explicitly configured and limiting other types of access for this account as necessary (cf. the difference between pilot and production roles).

The gLExec utility is activated and uses the user's proxy obtained from the MyProxy server. Effectively, each task is sandboxed by being executed under the end user account to which the user's proxy is mapped, with privileges limited to those of that account and proxy.

The identity switch via gLExec leads to each user's processes running under specific UIDs traceable to their respective identities. At the PanDA level, the DN of the job submitter is recorded permanently for each PanDA job, such that PanDA can trace and account usage.

When the retrieved credentials do not carry VOMS attributes, or they have expired, they need to be (re)added on the worker node by invoking the VOMS client. It is preferable that the delegated credentials already include the desired VOMS attributes (even if expired) to avoid an improper escalation of VOMS privileges in case a credential is compromised.

Any error during the execution of the payload is propagated back to the central data base, and reported through the PanDA monitoring web interface. The report includes both an error code and a meaningful message.

Finally, once the payload has been executed and the running process returns back to the pilot, the retrieved credentials are deleted from the local disk.

3.1. Issues related to the identity switch

Several issues surround the identity switch. The first one is that the running process is automatically assigned a home directory according to the new UID, and moved to it. The location and properties of this directory depend on the particular configuration and policies of the local batch system. The directory does not contain the files prepared by the pilot. The PanDA gLExec interface records the location of the working area and the first action after the identity switch is to go back to the pilot working area.

The properties of this pilot working area may vary depending of the local batch system. However, it is common that this directory and possibly a number of leading directories belong exclusively to the pilot. This implies the switched identity may not be allowed to create new files underneath this path, or even execution could be prohibited. To avoid this difficulty the PanDA gLExec interface changes the UNIX permissions of the working directory before performing the change of identity to allow the new one to execute and write. The original set of UNIX permissions is recorded and reestablished just after the gLExec invocation.

Another important issue is that the whole pilot environment effectively "vanishes" when the identity changes. This environment contains relevant pieces for the execution like the location of the libraries and similar information. The implemented solution consists of the reconstruction of the whole set of environment variables before continuing with the activities. An intermediate wrapper is created dynamically by the pilot, with the complete list of variables. This intermediate wrapper is the application run under gLExec. After the proper environment is thus recreated, the payload is finally executed from this wrapper.

In parallel, a second process is launched to renew the end user credentials periodically from the MyProxy server in case their life time is shorter than the payload execution duration.

3.2. Issues related to data movement

The payload execution can be seen as composed of three steps: input files stage-in, the user job execution, and output files stage-out. Whether to perform the three steps under the end user

identity or just the calculation part, while leaving the pilot to deal with the data movement, will depend on the policies of each site. We have made provisions to enable the PanDA gLExec interface to operate properly in both scenarios.

4. Deployment

The most important requirement to follow (by design) in order to deploy gLExec in an Identity-mapping model is that the installation has to be entirely local. No files can reside on distributed file systems. The executable and configuration files have to be owned by the superuser.

Deployment of gLExec in OSG sites requires also worker nodes to have a host certificate, which is used in the call to GUMS[14] to get the mapping. In this way, the communication between the client and the GUMS server is secured with the client host certificate. The subject of this certificate also indicates the worker node for which the possibly host-dependent mapping is to be determined. This certificate location is specified in /etc/glexec/contrib/gums/gums_interface.

Other components to be installed and deployed on the worker nodes to achieve all steps described are the VOMS client configuration (a "vomses" file or directory) and the MyProxy client software to allow the user credentials retrieval.

When a given site protects its network traffic with firewalls, communication between the worker node and the MyProxy server may not be allowed. In these cases a proxy service has to be deployed within the site boundary, the name of the MyProxy service points to this inner proxy, which redirects the communication to the real MyProxy server instance.

In practice, gLExec has been deployed and successfully tested on several OSG sites in the US (Brookhaven National Laboratory, Fermi National Laboratory and SLAC National Accelerator Laboratory). and a dedicated MyProxy server has been installed at Brookhaven National Laboratory.

The deployment of gLExec on the EGEE infrastructure is closely tied to the deployment of SCAS, the Site Central Authorization Service, which decides and records the mappings of proxies presented by gLExec to ensure that all worker nodes at the site map a given proxy the same way. The combined deployment of gLExec and SCAS has entered the limited preproduction stage on April 27 and is expected to reach the full production stage still in the spring of 2009. In parallel there have been a few production sites that have made gLExec and SCAS available as an experimental service, allowing ATLAS and other LHC experiments to develop and test the adaptation of their middleware to gLExec. It may be a few months before all EGEE sites relevant to ATLAS will have upgraded and enabled gLExec as desired. In the meantime PanDA would make use of gLExec where it is available.

Acknowledgments

The authors would like to thank Jim Basney (NCSA/OSG), Oscar Koeroo, Gerben Venekamp and David Groep (NIKHEF) for their constant support, feedback and assistance that made this project possible. We are also grateful to the site administrators at the sites in both the EGEE and OSG Grid projects for their effort with software installation and configuration, as well to the EGEE team in charge of the gLExec/SCAS deployment (Antonio Retico and Gianni Pucciani). This work was supported by the US Department of Energy and National Science Foundation, and managed by the Open Science Grid Consortium.

References

[1] LHC Computing Grid Project *http://www.cern.ch/lcg/*
[2] ATLAS Collaboration 1994 ATLAS Technical Proposal *CERN/LHCC/94-43*
[3] Foster I, Kesselman C and Tuecke S 2001 The Anatomy of the Grid: Enabling Scalable Virtual Organizations *International J. Supercomputer Applications* **15**(3)
[4] Enabling Grids for E-sciencE *http://www.eu-egee.org/*
[5] Nordic DataGrid Facility *http://www.ndgf.org/ndgfweb/home.html*

[6] Open Science Grid *http://www.opensciencegrid.org/*
[7] Nilsson P, Caballero J, De K, Maeno T, Potekhin M and Wenaus T 2008 The PanDA system in the ATLAS experiment *Accepted for publication in PoS, Proceedings of ACAT 2008 Conference.*
[8] Tuecke S, Welch V, Engert D, Pearlman L and Thompson M Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile *RFC* 3820
[9] Groep D, Koeroo O and Venekamp G 2008 gLExec: gluing grid computing to the Unix world *J. Phys.: Conf. Series* **119** 062032
[10] Groep D, Koeroo O and Venekamp G Grid Site Access Control and Credential Mapping to the Unix domain *Nikhef PDP Technical Report http://www.nikhef.nl/grid/lcaslcmaps/*
[11] Basney J, Humphrey M and Welch V 2005 The MyProxy Online Credential Repository *Software: Practice and Experience* **35, Issue 9** 801-16.
[12] pAthena *https://twiki.cern.ch/twiki/bin/view/Atlas/PandaAthena*
[13] Virtual Organizations Membership Service (VOMS) *http://www.globus.org/grid_software/security/voms.php*
[14] Hover J, Packard J et al 2004 Grid User Management System (GUMS) *https://www.racf.bnl.gov/Facility/GUMS/1.3/index.html*